UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

LEARNING FROM INTERPRETING TRANSITIONS IN EXPLAINABLE DEEP LEARNING FOR BIOMETRICS

Máster Universitario en Métodos Formales en Ingeniería Informática

Autor: ZilongWang

Directores: Ortega de la Puente, Alfonso
Ribeiro, Tony

Madrid
Septiembre de 2020

Acknowledgement

First of all, I would like to thank Prof. Alfonso Ortega De La Puente for his great help during this research. Although I had a little communication barrier due to language problems, he still patiently communicated with me weekly, explained the basic concept of logic programming and provided abundant useful advice for my research with his expertise and knowledge. Without his encouragement and continuous support, I cannot successfully complete my studies. At the beginning of this research, he arranged the tasks clearly in details, which made the entire process clear and rigorous. I owe a debt of gratitude to him not only for his support on my research, but also for sharing with me his attitude towards life. He kept encouraging me, strengthened my confidence and made me feel warm in this foreign country. He is the best teacher I have ever met.

In addition, I also want to express my gratitude to tutor Dr. Tony Ribeiro because he always shares his knowledge on this area. It was his serious working attitude, detailed comments, patient guidance and strong support that my research could be successfully completed. He made great contribution to this research. I appreciate his rigorous and serious attitude towards my thesis.

At last, I would like to thank my family for their selfless support. Without their sacrifice, I could never complete my research.

Abstract

Background: With the rapid development of machine learning algorithms, it has been applied to almost every aspect of tasks, such as natural language processing, marketing prediction. The usage of machine learning algorithms is also growing in human resources departments like the hiring pipeline. However, typical machine learning algorithms learn from the data collected from society, and therefore the model learned may inherently reflect the current and historical biases, and there are relevant machine learning algorithms that have been shown to make decisions largely influenced by gender or ethnicity. How to reason about the bias of decisions made by machine learning algorithms has attracted more and more attention. Neural structures, such as deep learning ones (the most successful machine learning based on statistical learning) lack the ability of explaining their decisions. The domain depicted in this point is just one example in which explanations are needed. Situations like this are in the origin of *explainable AI.* It is the domain of interest for this project. The nature of explanations is rather declarative instead of numerical. The hypothesis of this project is that declarative approaches to machine learning could be crucial in explainable AI.

Objectives: There are two main objectives of this study. The first one is to research all the inductive logic programming (ILP) algorithms which could learn a logic program from the large scale of data and its decisions made by machine learning algorithms. The second one is to select the most adequate ILP to reason about the biases of the machine learning algorithms.

Method: The unfair machine learning algorithm we concern is about an automatic recruitment system. The dataset contains 24,000 profiles, each CV contains 14 attributes and one unbiased score and two biased scores (gender and ethnicity). We learn logic programs from the dataset for different scenarios using an ILP algorithm called GULA-PRIDE. Then, we extract the biases from the logic programs learned from the unbiased data and biased one.

Results: For most of the scenarios we test, the GULA-PRIDE could output a logic program in less than one minute with high accuracy. These programs are declarative descriptions of the process. They are logically equivalent to the system on the set of inputs used. The conditions of each rule are guaranteed to be minimal. A naïve analysis of the programs allows to explain important characteristics of the datasets used. The most relevant is that when comparing the programs learnt for biased and unbiased datasets, we can easily observe this bias in the programs.

Conclusions: We select an efficient ILP algorithm GULA-PRIDE which could learn a logic program from a large scale of data efficiently. This system shows its capability and adequacy for generating relevant declarative explanations about what is going on in the process. For example, it is able to explain the structure of the dataset used and catches the bias when it is present.

Keywords: Inductive logic programming, Artificial Reasoning, Unfair Automatic Recruitment System

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

The framework of this research is exploring the feasibility of providing explanations for statistical machine learning approaches using GULA-PRIDE, and to what extent can GULA-PRIDE extract the bias of the unfair statistical machine learning approaches. In this chapter, we first outline the research topic. Secondly, it involves the field of research. Then, the research method are presented separately. Finally, we introduce the structure of this report.

1.1. Overview

Statistical machine learning algorithms have achieved great success in various applications such as speech recognition [Senior et al., 2012], image classification [He et al., 2016], machine translation [Wu et al., 2016] and so on. Among these approaches, deep neural networks have shown the most remarkable success, especially in the research area of facial recognition [Wu et al., 2016]. Although they usually have good generalization ability on similarly distributed new data, they share the weaknesses of the lack of explanations of the learning process and the poor understandability by humans of the whole process.

Another shortage of these machine learning algorithms is that they rely on data collected from society, and therefore may reflect or even amplify the biases if appropriate measures are not taken. [Acien et al., 2018; Drozdowski et al., 2020; Nagpal et al., 2019].

On the other hand, logic programming is a declarative programming paradigm with a higher level of abstraction because it is based on one of the formal models (first order logic) used to represent human knowledge. ILP has been developed for inductively learning logic programs from examples [Muggleton, 1991]. Roughly speaking, given a collection of positive, negative examples and background knowledge, ILP systems learn a set of declarative programs [Muggleton et al., 2014; Copper et al, 2019], which could even be noise tolerance [Dai et al., 2015; Muggleton et al., 2018b], that entails all of the positive examples but none of the negative examples.

For this project, learning from interpreting transitions (LFIT) [Ribeiro, 15; Ribeiro et al., 18] is one of the most promising approaches. LFIT learns a logic representation (digital twin) of any dynamical complex system by observing its behavior as a black box. The most general of LIFT algorithms is GULA (General Usage of LFIT Algorithm). PRIDE is an approximation to GULA with polynomial performance.

In this research, we start from an unfair automatic recruitment system from [Peña et al., 2020] in which two different biased datasets (gender and ethnicity) are tested. We apply GULA-PRIDE to these two different datasets. The object is to see if we can generate an explanation for the behavior of this unfair automatic recruitment system. Furthermore, we try to extract the bias of the decisions made by the model of the biased machine learning system with the help of GULA-PRIDE.

1.2. Field of Research

So far, learning from interpretation transition (LFIT) [Inoue et al., 2014] has been proposed to automatically learn a model of the dynamic system from the observation of its state transitions as in Fig. 1.1. To date, the following systems have been tackled: memory-less consistent systems [Inoue et al., 2014], systems with memory [Ribeiro et al., 2015], non-consistent systems [Martínez et al., 2015] and their multi-valued extensions [Martínez et al., 2016].

Fig. 1. The framework of LFIT

GULA is the generalization of LFIT that can learn digital twins of any system no matter the kind of its dynamics. In this research, we focus on a variant of GULA, called PRIDE that gets the same as GULA with polynomial performance. We apply it to generate an explanation for the behavior of the machine learning algorithms, and extract the bias of the unfair machine learning algorithms.

1.3. Research Method

The goals of the research are to answer two questions:

- Is it possible for GULA-PRIDE to explain what the deep learning process (automatic recruitment system) does?

- Is GULA-PRIDE able to catch the bias of the deep learning process (automatic recruitment system)?

To answer these two questions, we focus our research on a well-known problem: Fair Automatic Recruitment system. We slightly modify (discretization and reconstruction) the FairCVdb from [Peña et al., 2020]. Eight scenarios of experiments are designed from the dataset including biased and unbiased ones for gender or ethnicity. Then we apply GULA-PRIDE to the datasets to learn logic programs from these datasets, which could answer the first question. Then, we define some statistical variables for the logic programs to reveal the bias of the original data. These variables and the comparison of variables for different scenarios could completely answer the second question.

1.4. Work Structure

The structure of this report is shown below:

- Chapter 1 introduces the research work, which includes the research topic, purpose and method.
- The related works of different machine learning algorithms, including statistical approaches, discrete approaches and hybrid approaches, are presented in detail in Chapter 2. A brief introduction to the biometric recognition system and automatic recruitment system is also presented in this chapter.

- In Chapter 3, we applied GULA-PRIDE to the automatic recruitment system. Then we analyze the logic programs learned by GULA-PRIDE, and extract the biases from these logic programs.
- Chapter 4 draws a conclusion of the whole research and proposes possible future works.
- Chapter 5 gives the future research lines for this research.
- References.
- Appendix A gives an example of a logic program learned by GULA-PRIDE in our experiments.

2. LITERATURE REVIEW

This chapter presents some literature reviews of our research topic. Firstly, we show the state of art of the explainable machine learning approaches and biometric machine learning systems, especially the fair automatic recruitment systems. Then, we introduce some advances in the area of learning from transitions in detail as it's an important tool in this research.

2.1. State of the art of approaches that could be used for explaining deep learning processes

We will devote this section to the state of the art of those approaches that can be useful in explainable AI. To discuss these systems, we have to review different approaches to machine learning. By one side, deep learning is being considered a candidate approach to explain itself. By the other side there are declarative approaches to machine learning that from their own nature directly learn explanations.

The structure of this discussion will review attempts to explain deep learning from statistical-numerical machine learning approaches, from declarative approaches and for hybrid approaches.

The approach under consideration in this project belongs to the declarative ones.

2.1.1. Relevance of explainable machine learning algorithms

In the 1980s, a lot of machine learning models were proposed. To better understand the ability of these models, an operational criteria was provided by Michie [Michie, 1988]. In this work, three criterias were proposed: weak, strong and ultra-strong. Michie's aim was to evaluate the machine learning models with not only the predictive accuracy but also the comprehensibility of learned knowledge.

Michie's weak criterion identifies the models in which the machine learner is able to improve the predictive performance with increasing amounts of data. To fulfill his strong criterion, the machine learner should additionally provide its hypotheses in symbolic form. For the ultra-strong criterion, it strengthens the strong criterion by demanding the machine learner with the ability to generate new knowledge about the learnt concepts that could improve the performance of a human being provided with its learned hypothesis.

According to Michie's criterion, most of modern statistical machine learning models are consistent with Michie's weak criterion because the only explanation usually available about their decisions is that it chooses the highest probability among the possible. To experimentally demonstrate Michie's ultra-strong machine learning criterion, Schmid et al. [Schmid et al., 2017] introduced an operational definition for comprehensibility of logic programs. Then, Muggleton et al. [Muggleton et al., 2018a] showed that Inductive logic

programming (ILP) which includes approaches able to automatically get logic programs from a set of examples and counterexamples is consistent with Michie's weak criterion.

This old classification has recently gotten the focus again due to the impressive increase of performance and successful use in different domains of machine learning techniques from the emergence of deep learning.

More abstract and comprehensible explanations than comparing the values of the corresponding probabilities are always welcome. But some new problems arose when, for example, machine learning approaches are used in automatic recruitment of candidates by the companies [Fiérrez et al., 2017a; Fiérrez et al., 2017b] or authorizing banking products (https://www.bbc.com/news/business-50365609) for identifying criminals in forensic scenarios. In these domains, explanations are mandatory to ensure the needed guarantees. The automatic procedure has to be ethical and be bias free by, for example, gender and ethnicity [Peña et. al., 2020]. The term explainable algorithm has been coined for referring to all those approaches that try to move numerical-statistical machine learners (usually weak) to the strong and ultra-strong group.

But an explainable machine learning algorithm is not just adding obvious labels or rules to simplified versions of the learning process. Great amounts of effort are being devoted to identify and warn the community about pitfalls that should be avoided when explaining machine learners [Molnar et al., 2020].

More detailed description about the state of the art of explainable machine learning algorithms (mainly from numerical- statistical approaches) can be found in text like [Herrera et. al, 2019] or [Molnar et al., 2020]

2.1.2. Relevance of explainable machine learning algorithms from formal methods

If you ask a group of persons about machine learning most of them will create in their minds the image of a neural structure, some of them a deep pipe system of interconnected neural levels, some of them a single perceptron. It is true that these kinds of systems are the style now; but it is unfair. Statistical regression, for example, is a classical machine learner (at least strong, by the way). But there exists also a powerful approach to ultra-strong machine learning from declarative approaches around the idea of ILP. This term is also abusing the meaning of "logic" because it also is applied to other declarative paradigms. It stands for all the approaches that automatically learn declarative programs from a set of examples and counterexamples. It has been devoted a great effort for the last two decades to ILP. In this case declarative programs mean programs written with declarative programming languages as Prolog (full first order logic or some approaches rather propositional logic), Datalog, answer set programs, the functional program Haskell [Katayama, 2005], etc. ILP has a promising future in a wide range of domains some of them explainable machine learning algorithms [Cropper et al., 2020]

In the following section we will argue that ILP and other declarative approaches are ultra-strong machine learners. Their theoretical basics (mainly logical) allow them to benefit from results available in this realm: to learn from a much smaller set of examples, or even guarantee that the learn program is the smallest (or the optimum once a logical description of the cost criteria is provided). You can see some of these characteristics in [Cropper, 2017; Cropper et al., 2020].

2.1.3. Numerical statistics approaches useful for explainable AI

The first attempts to incorporate explanations to machine learning appear in the machine learning realm that needs them. As it has been previously pointed, the best known and currently more popular machine learning systems are based on numerical treatment of statistical techniques. Some approaches try to get explanations by interpreting what is learned in intermediate levels of the system (for example in deep learning systems [Kanehira et. al, 2019]) A detailed description of these attempts can be found in [Herrera et al., 2019]
They try to take advantage of the characteristics of these kinds of machine learners: high accuracy if a great number of examples is available, noise tolerance and great generalization capacity.

2.1.4. Discrete/formal approaches useful for explainable AI

The hypothesis of declarative approaches to explainability is that the nature of the explanations is more declarative than numerical. It seems, from this viewpoint, that declarative approaches may be more adequate for getting explanations than statistical-numerical approaches.

2.1.4.1. Metaheuristic (evolutive) models for strong machine learning

Evolutionary computation stochastically searches huge spaces of possible solutions as an alternative to brute-force approaches [Eiben et al., 2003]. The search is driven by mimicking the natural process of evolution and selection of species based on the survival of the fittest and the inheritance of the modified genetic mixture of the parent's genetic material. One critical component of these systems is the codification of candidate solutions. This process can be as simple as using a vector of bits (as in the original genetic algorithm proposal were used) or as complex as needed. In this way evolutionary computation has become a general purpose automatically programming paradigm because you can translate any problem into a genetic searching version.

So that, from a machine learning viewpoint, metaheuristic approaches like evolutionary computation, that find solutions to any kind of problem without providing by themselves any kind of explanation, could be considered weak machine learners.

But the most general and powerful proposal from the viewpoint of increasing the generality of evolutionary computation is genetic programming (namely automatically evolutionary programming). These systems use more sophisticated codification procedures in such a way that candidate solutions are interpreted as programs. Genetic programming originates with Koza's works [Koza, 92]. He proposed to codify solutions as trees, understood as LISP expressions. But recent approaches use more expressive formal models (context free, attribute and Christiansen grammars) to automatically program in any language (providing the grammar) and ensuring that all the solutions are syntactic and semantically correct [Ryan et al., 2003; Ortega et al., 2007]. An algorithm possible in a high-level programming language is an obvious symbolic representation of a task. This is why evolutionary automatic programming systems are strong machine learners.

2.1.4.2. Declarative models for ultra-strong machine learning

First order logic approaches

Logic programming is based on Robinson's resolution method [Lloyd, 87]. To program in this paradigm is actually to prove new logic formulas from the knowledge of a domain expressed in first order logic (Horn clauses).

This strategy of deduction based on resolution is quite similar to going across a graph using in each step the first order logic inference rule known as resolution. It aims to find the empty clause that is an alternate form to finish a demonstration by reducing ad absurdum (to get a contradiction from the hypothesis of negating the task expressed by the program, that is, by negating the formula that is being deduced). It is worth highlighting the role of the most general unification operation implied in each resolution step. Informally the most general unifier of two formulas is the most reasonable assignment of elements from the domain under consideration to the variables of the formulas for getting the same expression. Here, the "most reasonable" means the one that loses less generality.

One of the most powerful features of these programming languages is the possibility to incorporate to the theory contained in the original program any formula that can be deduced from it. This characteristic allows to develop programs that take as inputs a set of initial facts (the typical positive and negative examples of machine learning algorithms) and get as output a set of clauses (a logic program) from which all the positive examples are deduced and none of the negative.

From a technical viewpoint there are some worth mentioning points: the ability of inventing new predicates and the expressivity of these new predicates (mainly if they are recursive or not). The state of the art of ILP, after more than two decades of continuous effort and development, offers systems that guarantee both things. With these features all these systems ensure to be able to learn any program for any computable tasks in a more or less general way.

The main goal of ILP as it has been described so far, considering as inputs only the sets of positive and negative examples is intractable because the set of all possible first order logic formulas (that is clearly infinite) should be potentially traversed.

In almost all real scientific domains there exists a previous knowledge from which new discoveries are made. This *background knowledge* can be expressed in ILP systems in the same formalism as the rest of the system (first order logic). Background knowledge is used in ILP to reduce the size of the set of formulas that has to be traversed to learn. Only those that use background knowledge and new predicates are considered.

But even with this help, in most of the real domains the set in which to look for is still too big and the learning process is still intractable. Formal properties of first order logic allows to normalize theories in such a way than with a small set of different structures (or schemes) for the clauses used as the bodies of the learnt predicates it is theoretically possible to automatically learn any computable algorithm. This bias in the structure of the clauses is called (among other names) meta rules.

Background knowledge and meta rules are enough to make the learning process tractable for most of the cases. Some systems (like Metagol [Cropper, 2017]) offer an ILP implementation that ensures to efficiently and automatically learn any computable task in any domain.

It is also possible to add to ILP mechanisms to measure the cost of the learnt programs.

These mechanisms are translated into logical clauses as the rest of the ILP components. The costs can be for example, in terms of resources required or complexity of the learnt programs. In this way it is possible to extend ILP to efficiently and automatically learn efficient programs for any computable task in any domain. Metaopt [Cropper, 2017] is an implementation of one of these systems.

Answer set programming approaches

Other approaches (such as the answer set programming - ASM- paradigm [Gebser et al. ,2012]) tackle this same goal from transforming the first order logic theory under the examples into a propositional one by the process known as grounding.

Grounding consists of replacing each variable by all the possible non variable objects relevant for the demonstration (that is in the worst case the Herbrand base).

After grounding you get a propositional version of the initial first order theory but only on the relevant objects. On this propositional theory all the inference rules of the propositional logic are applicable.

Proofs become in this way in a two steps process: to generate all the ground atoms - predicates applied to non-variable arguments- compatible with the original theory and examples; and to compute the models -that really are functions that assign constant objects to variables- that satisfy them. In these paradigms, deduction is actually made by filtering the relevant objects after grounding generates all of them. Grounding obviously implies a potential combinatorial explosion. These paradigms provide the programmer with mechanisms to reduce it the most trying to generate only those combinations really relevant for the solution of the problem.

Once we have explained the different context of these paradigms, we are able to see that they can face the same goals than ILP with similar methods: the same formalism is used both to describe positive and negative examples and the program that will be automatically learnt. And the same arguments used in ILP explain in these paradigms, the use of a “theory” as background knowledge and some mechanism to bias the space of possible programs to those more promising. Notice that in ASP the term theory may be too strong because it is rather an answer set program. [Law, 2018] describes ILASP, a system that implements this approach.

Opportunity for hybridization

It became clear, from the previous explanation, that in ILP systems based on grounding to filter the relevant objects can be seen as a typical search problem in huge sets of candidates. These systems can benefit, therefore, from usual non-brute force search techniques, such as metaheuristics approximations (genetic algorithms, for instance). This is the case of GA-Progol [Alireza, 2013], for example.

Learning from interpreting transitions

There are some other approaches to ILP whose goal is the *explanation* itself of a given phenomenon. We mean that their aim is to automatically catch the semantics of the process under study and generate a logic version that behaves the same. This logical version is itself new knowledge about the process in a symbolic way that is the requirement to be considered an ultra-strong machine learner. These approaches usually name the learnt program “digital twin” of the system under consideration.

The input of some of these approaches is the description of the dynamics of the system expressed as a set of "transitions" (pairs of inputs-outputs). This description is translated into a propositional logic theory. From this theory a set of clauses is automatically learnt. The resulting program is logically equivalent to the system under consideration and the conditions of each clause are ensured to be minimal. This is the approach of learning from interpretation transitions: the family of algorithms by Dr. Ribeiro (co-advisor of this project) that includes LFIT, GULA (that is a generalization of LFIT to any kind of dynamics) and PRIDE that is an approximation with polynomial performance and that will be used in this project [Ribeiro, 2015; Ribeiro et al., 2018].

2.1.4.3. Hybrid approaches useful for explainable AI

Some other attempts try to benefit from the advantages of the approaches previously described (numerical and declarative): learning new knowledge (ultra-strong) from small sets of examples, noise tolerance, great accuracy and generalization.

Some approaches modify the ILP inference mechanism to make it differentiable and hence allow the use of techniques for reducing loss based on this differentiable version to learn. Loss usually is related with the number of examples not covered. It is usually needed to add numerical values to both facts and rules and mechanisms to compute these values while reasoning creates them. Some systems even connect different learners (for example, neural architectures are trained for recognizing inputs to the system with a numerical value that could be the probability produced by the learner. Deepproblog [DeRaedt et al., 2018] is one of these systems. Among the approaches that make the deduction process differentiable we can mention δILP [Evans, 2017], diff-log and prosynth [Mukund et al., 2019a; Mukund et al., 2019b].

Other approaches are based in probabilistic logic programming implementations: prob2foil [DeRaedt et al., 2015] is an extension to Problog that makes possible ILP under noise with predicate invention.

Other hybridizations extend ILP for handling noise and mistakes in the input examples but keeping the logical theoretical guarantees (for example iterating the ILP process on samples of the input to get the program with highest accuracy). Some of these approaches are ILASP (on answer set programs), Metagol-NT (on ILP) and GULA (from the learning by interpreting transitions viewpoint). All these systems are possible candidates to generate explanations in machine learning domains.

2.2. State of the art of machine learning systems and fair recruitment processes

Over the last decades, biometric machine learning systems have been applied to almost every aspect of our daily life, such as ad delivery systems [Ali et al., 2019], speech

recognition [Senior et al., 2012], image classification [He et al., 2016], machine translation [Wu et al., 2016] and so on.

In the area of facial recognition in forensic scenarios, machine learning algorithms are applied to different regions of the human face in various forensic scenarios (distance) [Tome et al., 2013]. Compared to traditional facial recognition systems, the accuracy could be highly improved. Soft biometrics (such as height) are also added to improve the performance of the whole system [Tome et al., 2014; Tome et al., 2015].

With the wide application of these machine learning systems, the shortage that the decision made by these systems could be unfair stands out. This problem is especially serious in the domain of the recruitment systems in which both the models and their training data are usually private for corporate or legal reasons. For example, Amazon's recruiting system is preferring male candidates over female candidates [Koditan, 2019].

To protect fairness, new regulations have been proposed by governments. Among these, the European Union's General Data Protection Regulation (GDPR)2 is especially relevant to constraint on the use of machine learning algorithms [Goodman et al., 2016]. The demand that the algorithm should provide an explanation for its decision. Although several research [Kim et al., 2019; Morales et al., 2019] efforts to construct a fair recruitment system, it is still a big challenge for the traditional recruitment system to provide an explanation.

2.3. State of the art learning from transitions

GULA (and PRIDE) is the system chosen in this project to see if it is possible to provide declarative explanations in real deep learning domains. The approach of learning from interpreting transitions was introduced in previous sections. In this section we will describe with some examples its approach.

GULA (and PRIDE) consider the system it has to learn described by means of

- The set of variables that are considered as inputs and outputs of the systems. They are usually discrete multivalued variables (although it is also possible to use continuous numerical variables until some extent)
- The set of transitions (pairs input-output) that describes the system as a black box.
    - Each transition is in fact a pair of two states each of which is a specific assignment of a given value to each variable (to all of them)

It automatically learns a logic program (set of clauses) that is logically equivalent to the system on the inputs provided. In addition, the conditions of every rule are minimal.
We will study this example:

```
if V={x0[0,1], x1[0,1,2], x2[0,1], y0[0,1,2,3], y1[0,1]}
```

Where

- x's variables are assumed to be inputs.
- y's variables are assumed to be outputs.

GULA (and PRIDE) generates multivalued logic (MVL) programs of the form

```
yj(value) <- xj1(vj1),..., xjm(vjm)
```

GULA consider that MVL rules are partially and asymmetrically ordered by the domination relationship (rules with the most general bodies dominates those with less general):

R1 dominates R2, written R2 ≤ R1 if h(R1) = h(R2) and b(R1) ⊆ b(R2)

For Example

```
r1: y0(1)<- x1(0),x2(0)
r2: y0(1)<- x0(0),x1(0),x2(0)
r1 dominates r2 because they have the same head and the body of r1 is
included (or equal) to the body of r2.
```

A key concept is to match a rule to a state (represented by the operator ⊓). This happens when its body is included in the state

```
Given a state s: {x0(0), x1(1), x2(1), y0(0), y1(0)}
and a rule r: y0(2)<-x1(1)
r⊓s because of x1(1)
```

This operation is used to define when a rule or a program (set of rules) realize a transition:

- When both the head and body of the rule belong (respectively) to s' and s.
- The program P realizes (s,s') when for every variable you can find a rule R in P (with the var(head(R)) thata variable) that realices (s,s')

```
Given a transition
t:(   s1: {x0(0),x1(1),x2(0),y0(1),y1(0)},
      s1':{x0(1),x1(0),x2(1),y0(0),y1(1)},
and a rule r: y0(0)<-x1(1),x2(0)
r realizes t because of y0(0) and

Given the program
P={
      r1: x0(0) -> x0(1)
      r2: x0(0),x1(0) -> x1(0)
      r3: x2(0)-> x2(1)
      r4: x0(0),x2(0)->y0(0)
      r5: x1(1)->y1(1)}
And the transition
t1:(  s1: {x0(0),x1(1),x2(0),y0(1),y1(0)},
      s1':{x0(1),x1(0),x2(1),y0(0),y1(1)},
P realizes t1 because for each variable you can find at least one rule
that matches t1 which gives a value to this variable as head
r1 has x0; r2 x1; r3 x2; r4 y0 and r5 y1
r1 realizes t1 because of x0(0) in s1 and x0(1) in s1'
r2 realizes t1 because of x0(0), x1(1) in s1 and x1(0) in s1'
r3 realizes t1 because of x2(0) in s1 and x2(1) in s1'
r4 realizes t1 because of x0(0), x2(0) in s1 and y0(0) in s1'
r5 realizes t1 because of x1(1) in s1 and y1(1) in s1'
Taking into account that x's are inputs and y's are outputs P makes
only sense if it includes only r4 and r5.
```

What happens when some rule does not make sense; that is, its body is coherent with data but its conclusion is not? GULE names this situation conflict. A rule R can conflict with a set of transitions T if there exists a state s in the set of the states that are in the first position of the of the transitions of T (fst(T)) for which R matches (its body is included in s) but its head does not belong to any second part of T. In other words: for a rule to be in conflict, her body should match some state of T. Intuitively the rule is saying something is possible

from an observed initial state but actually the conclusion is never observed so the rule is wrong.

```
Give the set of transitions
T= {  t1,..,ti,
      ti+1: (
                  s1: {x0(1),x1(1),x2(1),y0(1),y1(1)},
                  s1':{x0(0),x1(0),x2(1),y0(1),y1(0)}
            ),
      ti+2: (
                  s1: {x0(1),x1(1),x2(1),y0(1),y1(1)},
                  s2':{x0(1),x1(1),x2(1),y0(2),y1(1)}
            ),
      ti+3: (
                  s1: {x0(1),x1(1),x2(1),y0(1),y1(1)},
                  s3':{x0(0),x1(2),x2(1),y0(3),y1(0)}
            ),
      ti+4: (
                  s1: {x0(1),x1(1),x2(1),y0(1),y1(1)},
                  s4':{x0(1),x1(2),x2(1),y0(3),y1(1)}
            ),
      ti+5,...,tn
}
Where the transitions ti+1,...,ti+4 are all the transitions in T that
have as first state s1
And the rule
r: x1(1), x2(1)->y0(0)
No matter what happens with the transitions not fully described
s1 exists and the body of r ( x1(1),x2(1) ) is included into s1 in
transitions ti+1,...,ti+4 and, in addition its head (y0(0)) does not
belong nor to s1', nor s2', nor s3', nor s4'.
We can conclude that r conflicts with T
```

Several rules can be applicable at the same time.

- Rules R and R' concurrent with respect to a set of states S
    - When there exists a state s in S for which R and R' matches (that is their bodies are included in s)
    - The heads of the rules have the same variable
    - But they (the heads) differ

For programs it is important to know when they are complete and consistent:

- A program P is complete with respect to a set of states S and variables V
    - For all states s and variables v
    - P contains a rule R
        - That matches s
        - Its head has as variable v
    - Notice that R does not realices any transition, it just ensures to mach s and its head talks about v for any combination

- A program P is consistent with respect to a set of transitions T
    - If P does not contain any rule conflicting with T

The learning operations defined to induce the logic program equivalent to the observed transitions are

- The least specialization of a rule R and a state s

- That really is the new rule R' that extends body(R) with
  - those variables not in body(R)
  - with the values that are not in s
- Notice that if a rule does not match any observation, it is still consistent since no counterexample is observed.

Example (in this case we are using taking apart inputs from outputs):

```
if V={x0[0,1],x1[0,1,2],x2[0,1],y0[0,1,2,3],y1[0,1]}
R: y0(1)<-x0(1),x1(2)
And s:{x0(1)x1(1)x2(1)}
The v not in body(R) is x2 (excluding outputs)
The values not in s for x2 are 0
So the least specialization of R and s is R': y0(1) <-x0(1), x1(2),
x2(0)
```

Take into account that it does not produce only one rule but as many as free conditions in the body. If there are 10 variables and R has already 2 conditions, there will be 8 least specializations. This is a point in which GULA and PRIDE differ: GULA will consider all of them, PRIDE only one.

- The least revision of a program P and a state s
  - Take the set of transitions that has as first state s
  - Remove from P those rules that conflicts with T
  - But add for each of them each least specialization on s

```
Let's consider the program
P={
      r1: x1(1)x2(1)->y0(0)
      r2: x0(0)->y1(0)
      r3: x0(1)x2(1)->y0(3)
      r4: x2(0)->y1(1)
}
And the set of transitions
T= {
      t1_1: (
                  s1: {x0(1),x1(1),x2(1),y0(1),y1(1)},
                  s1':{x0(0),x1(0),x2(0),y0(1),y1(0)}
            ),
      t1_2: (
                  s1: {x0(1),x1(1),x2(1),y0(1),y1(1)},
                  s2':{x0(1),x1(1),x2(1),y0(2),y1(1)}
            ),
      t1_3: (
                  s1: {x0(1),x1(1),x2(1),y0(1),y1(1)},
                  s3':{x0(0),x1(2),x2(0),y0(3),y1(0)}
            ),
      t1_4: (
                  s2: {x0(0),x1(0),x2(0),y0(0),y1(0)},
                  s4':{x0(1),x1(2),x2(1),y0(3),y1(1)}
            ),
      t2_1: (
                  s2: {x0(0),x1(0),x2(0),y0(0),y1(0)},
                  s5':{x0(_),x1(_),x2(_),y0(1),y1(1)}
            ),
      t2_2: (
                  s2: {x0(0),x1(0),x2(0),y0(0),y1(0)},
                  s6':{x0(_),x1(_),x2(_),y0(2),y1(1)}
            ),
      t2_3: (
```

```
                    s2: {x0(0),x1(0),x2(0),y0(0),y1(0)},
                    s7':{x0(_),x1(_),x2(_),y0(3),y1(1)}
                )
}
```

In this case we are taking apart inputs from outputs.
R1 conflicts with T because:

- There exists a state s1 in fst(T) ({x0(1), x1(1), x2(1), y0(1), y1(1)}) that contains its body (x1(1)x2(1))
- But its head (y0(0)) is not contained in any of the second states of the transitions with s1 as first state.

While rule R3 does not because having the same body contained in s1, its head (y0(3)) is contained at least in one of the second states of the transitions with s1 as first state (more specifically t1_4).

By similar reasons R2 conflicts with T because:

- There exists a state s2 {x0(0), x1(0), x2(0), y0(0), y1(0)} that contains its body (x0(0))
- But its head y1(0) is not contained in any of the second states of the transitions with s2 as first state.

By the other hand rule r4 does not conflict with T because its body (x2(0)) is only contained in s2; and its head (y1(1)) is included in all the transitions with s2 as the first state.

Following the preceding algorithm, we can compute:

```
Lpse(r1,s1)={x1(1),x2(1),x0(0)->y0(0)}
```

- Because (excluding outputs) the variable not in the body of r1 is x0 and the only value of x0 not in s is 0

Lpse(r2,s2)={x0(0),x1(1)->y1(0), x0(0),x1(2)->y1(0), x0(0),x2(1)->y1(0)}

- Because (excluding outputs) the variables not in the body of r2 are x1 and x2 and the values of x1 not in s2 are 1 and 2; and the value of x2 not in s2 is 1.

From this we can finally compute:

```
Lrev(P,T)=(P−{r1,r2}) ∪ Lspe(r1,s1) ∪ Lspe(r2,s2)=
    { r3: x0(1)x2(1)−>y0(3), r4: x2(0)−>y1(1) } ∪
          {x1(1)x2(1)x0(0)−>y0(0)} ∪
{x0(0)x1(1)->y1(0), x0(0)x1(2)->y1(0), x0(0)x2(1)->y1(0)}
```

These operations allow us to realize that the “desired program” we are looking for (consistent with the observed transitions, that realizes them, and whose rules have irreducible conditions) is unique and can be obtained by the reiterated application of the operations we have just described.

The exact procedure to learn the program is to iteratively do that

- Remove all the rules that matches with negative examples
- Least revision of P and the set of transitions
- Remove from P rules dominated

- Starting from the transitions observed considering the other combinations as negative examples

3. PROJECT CONTRIBUTIONS

In this chapter, we describe the main work we have done. Firstly, we introduce the motivation of the research which could be summarized as two questions. The rest of the chapter shows the design of experiments and the analysis of the results focusing on these two questions.

3.1. Motivation

As we explained in Chapter 2, GULA-PRIDE could learn a logic representation of any dynamical complex system by observing its behavior as a black box. In this research, we regard deep learning processes as a general dynamical complex system and apply GULA-PRIDE to the input and output of the deep learning system. From the learned logic program of GULA-PRIDE, we could acquire declarative explanations of the deep learning process.

In this research, we focus on a well-known problem from colleagues of our department: Fair Automatic Recruitment system. As we have previously mentioned; in their previous work [Peña et al., 2020], they showed that the decisions made by automatic recruitment algorithms may reflect current and historical biases. They showed that the distributions of scores for different gender or ethnicity are unbalanced when the automatic recruitment system is trained with biased data. For us, we try to figure out two questions:

- Is it possible for GULA-PRIDE to explain what the deep learning process (automatic recruitment system) does?
- Is GULA-PRIDE able to catch the bias of the deep learning process (automatic recruitment system)?

The rest of this Chapter is focused on these two questions. To answer these two questions, we apply GULA-PRIDE to the dataset mainly from [Peña et al., 2020] and analyze the learned logic programs in two dimensions.

3.2. Experiments performed

In order to evaluate to what extent GULA-PRIDE could catch the semantics of the process, we slightly modify the FairCVdb from [Peña et al., 2020], which contains 24,000 synthetic resume profiles including 12 features obtained from 5 information blocks (education, availability, experience, the existence of a recommendation letter and language proficiency in a set of 8 different languages), 2 demographic attributes (gender and ethnicity), a face photograph and two scores (biased and unbiased). For our case, we discard the face photographs. Also, to make it suitable for GULA-PRIDE, we need to discretize the value of every attribute.

The FairCVdb is divided into two subsets: Test and Train. The unbiased scores for both subsets are fixed manually from the 5 information blocks, based on a formula consulted with a human recruitment expert. The biased scores for Train are generated by applying a

penalty factor to certain individuals belonging to a particular demographic group while the biased scores of Test are decided by automatic recruitment system trained with biased data from Train.

To better utilize the data, we divide our experiments into 8 scenarios:

- Scenario 1: Inputs: gender, "other inputs" / Outputs: score without bias from Test dataset;
- Scenario 2: Inputs: gender, "other inputs" / Outputs: score with bias from Test dataset;
- Scenario 3: Inputs: ethnicity, "other inputs" / Outputs: score without bias from Test datasets;
- Scenario 4: Inputs: ethnicity, "other inputs" / Output: score with bias from Test dataset;
- Scenario 5: Inputs: gender, "other inputs" / Outputs: score without bias from Tests and Train datasets;
- Scenario 6: Inputs: gender, "other inputs" / Outputs: score with bias from Tests and Train datasets;
- Scenario 7: Inputs: ethnicity, "other inputs" / Outputs: score without bias from Tests and Train datasets;
- Scenario 8: Inputs: ethnicity, "other inputs" / Output: score with bias from Tests and Train datasets;

In all these scenarios, "other inputs" means 2 to 12 attributes from the left 4 information blocks. So, totally, we have 88 individual experiments with GULA-PRIDE. Worthy to be mentioned that the first four scenarios are corresponding to the machine learning approach while the purpose of the last four scenarios are to test the ability of GULA-PRIDE with more complicated data.

We have decided to use test and test+train datasets because the aim of GULA-PRIDE is not to generate a predictor but a logically equivalent program to the system on the inputs provided. So, if we want to get explanations it seems to be useful to feed the system with as much data as possible. The experiments using only tests are for mimicking the behavior of the classifier learned on test data.

The program is run on a Xiaomi personal computer with Ubuntu 20.04 TLS 64-bit system, equipped with a 1.8GHz Intel Core 4 Quad CPU and 15.9GB RAM[1].

3.3. Results and discussion

For each of the experiments we performed, GULA-PRIDE outputs a logic program from the data and the accuracy for the logic program, which is calculated by comparing the original scores with the scores predicted by the logic program. We try to reply to the two questions we mentioned before based on this information.

3.3.1. First question

In this subsection, we try to answer this question:

[1] The learnt logic programs can be found here: https://drive.google.com/file/d/14LhTte_T-dXoG0lx_beR4LfaHZlb5Pbj/view?usp=sharing

- Is it possible for GULA-PRIDE to explain what the deep learning process (automatic recruitment system) does?

3.3.1.1. Analysis

To do this, we consider the 4 experiments with 13 inputs each for gender and ethnic. The basic information about these experiments are shown in Table 3.1.

Table 1.The number of rules of the logic program learned in experiments with 13 inputs and timing of the implements.

Scenario	gender				ethnicity			
	1	2	5	6	3	4	7	8
Number of rules	826	898	2200	2449	913	1350	2732	4183
timing (s)	2.402895	2.610074	36.614664	38.579469	2.512109	4.287454	43.826013	72.548077

As we can see in the table, GULA-PRIDE could finally provide logic programs in a reasonable time for all scenarios, which are declarative explanations for what the deep learning process (automatic recruitment system) does. Also, the accuracy for all of these experiments are more than 99.5%. It means that we can believe that our logic programs are corresponding to the behavior of the deep learning process with high confidence.

Compared to the deep learning approaches, the logic programs learned by GULA-PRIDE cannot be used to predict. These logic programs are logically equivalent to the data it learns from. If you test it for some fresh data, the accuracy will be very low similar to over-fitted models in deep learning. It is possible, nevertheless to use the programs learnt by GULA-PRIDE to predict but a post-processing stage on the rules if needed taking into account expert information on the domain. To get predictors from GULA-PRIDE is an open question under consideration by their author (as we know from personal communications and discussion So, as we can see in the table, the number of rules is depending on both the scale of examples and the logic complexity of the data. The first point can be verified with Scenario 1 (2, 3, 4) and Scenario 5 (6, 7, 8) and the second point can be verified with Scenario 1, 2 ,3 ,4 or Scenario 5, 6 7, 8.

Appendix A shows the output for one of these scenarios.
It is worth mentioning the question about the interpretability of the learnt programs. We conclude that the declarative explanation of what is going on in the process is clear. But its meaning could require the interpretation of an expert of the domain.

3.3.1.2. conclusion

Obviously, the answer to the first question is positive. The logic program is an explanation for the deep learning process. What's more, the high accuracy also shows that these logic programs are credible.
But we will highlight these questions:

- GULA-PRIDE is able to offer declarative explanations of what the behavior of the system (because the logical program learnt is logically equivalent to the system on the used inputs)
- In spite of the completeness of the explanations provided (they are in fact a program logically equivalent to the system) it is not useful as a predictor, that is, its behavior on fresh data is not guaranteed. For ensuring that the learnt program can be used for this purpose a post-process of the rules is needed.

3.3.2. Second question

In this subsection, we try to answer this question:

- Is GULA-PRIDE able to catch the bias of gender and ethnic directly?

For answering this question, firstly, we check if some values of the biased attributes (gender/ethnic) in the biased experiments with 13 inputs tend to get higher value than the others. In Analysis 2, we check if the learned logic programs show how the relevance of gender/ethnic is catched in biased versions vs non biased ones.

3.3.2.1. Analysis 1

We try to evaluate how and to what extent an automatic recruitment system algorithm is influenced by biases that are present in our dataset.

The intuition of this approach is simple: from the observation, we find that, in the biased experiments, $gender(0)$ appears more frequently in the rules lead to higher scores than $gender(1)$. It means that males tend to get higher scores than females. This phenomenon also exists in experiments for ethnicity. The objective of this analysis approach is to show this intuition quantitatively.

First of all, we define partial weight PW as follows.

For any program P and two atoms $v_0^{val_0^i}$ and $v_1^{val_1^j}$, where $val_0^i \in val_0$ and ${val_1}^i \in val_1$. Define $S = \forall R \in P \wedge v_0^{val_0^i} \in h(R) \wedge v_1^{val_1^j} \in b(R)$, then we have:

$$PW_{v_1^{val_1^j}}(v_0^{val_0^i}) = |S|.$$

The $PW_{v_1^{val_1^j}}(v_0^{val_0^i})$ corresponding to the frequency of $v_1^{val_1^j}$ for all the rules in with output $v_0^{val_0^i}$. A more accurate could be defined, for example, we could set a weight for rules with different length. But for our purpose, the frequency is enough.

Then, we define global weight GW depending on PW.

For any program P and $v_1^{val_1^j}$, we have:

$$GW_{v_1^{val_1^j}} = \sum_{val_0^i \in val_0} PW_{v_1^{val_1^j}}(v_0^{val_0^i}) \cdot val_0^i.$$

The $GW_{v_1^{val_1^j}}$ is a weighted addition of all the values of the output, and the weight, in our case, is the value of scores.

Take the logic program learned in Case 1 with 3 inputs as an example. The whole logic program is shown in Appendix A. In this case, v_0 and v_1 correspond to $scores$ and $gender$. The partial weights are shown in Table 3.2.

Table 2. partial weight for example logic program.

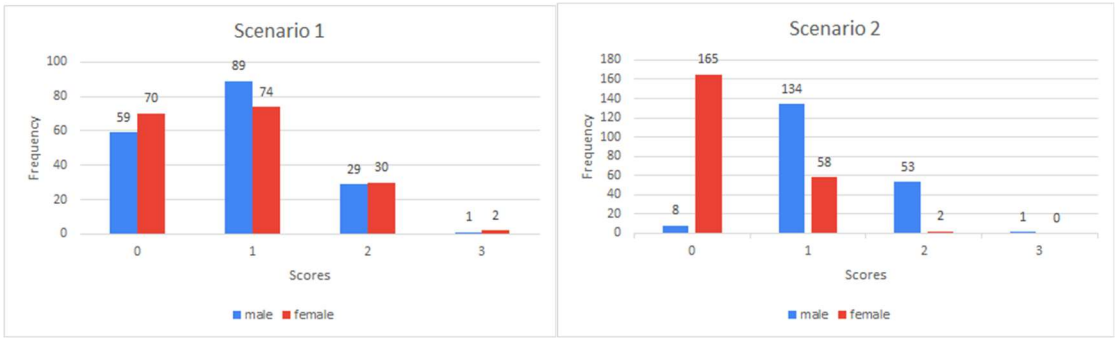
Scores / Gender	0	1	2	3
0	0	1	1	0
1	0	3	0	1

Then, we can easily calculate the group weight for the $gender(0)$ and $gender(1)$ as:

$$GW_{gender(0)} = 3;$$

$$GW_{gender(1)} = 6.$$

The analysis is based on the results of the 8 experiments with 13 inputs. Two dimensions are included in this analysis. Firstly, we consider the distribution of partial weight for each of the experiments. Then we calculate the global weight for different values of gender/ethnicity, and try to extract the bias from the distribution of the global weight by the comparisons.



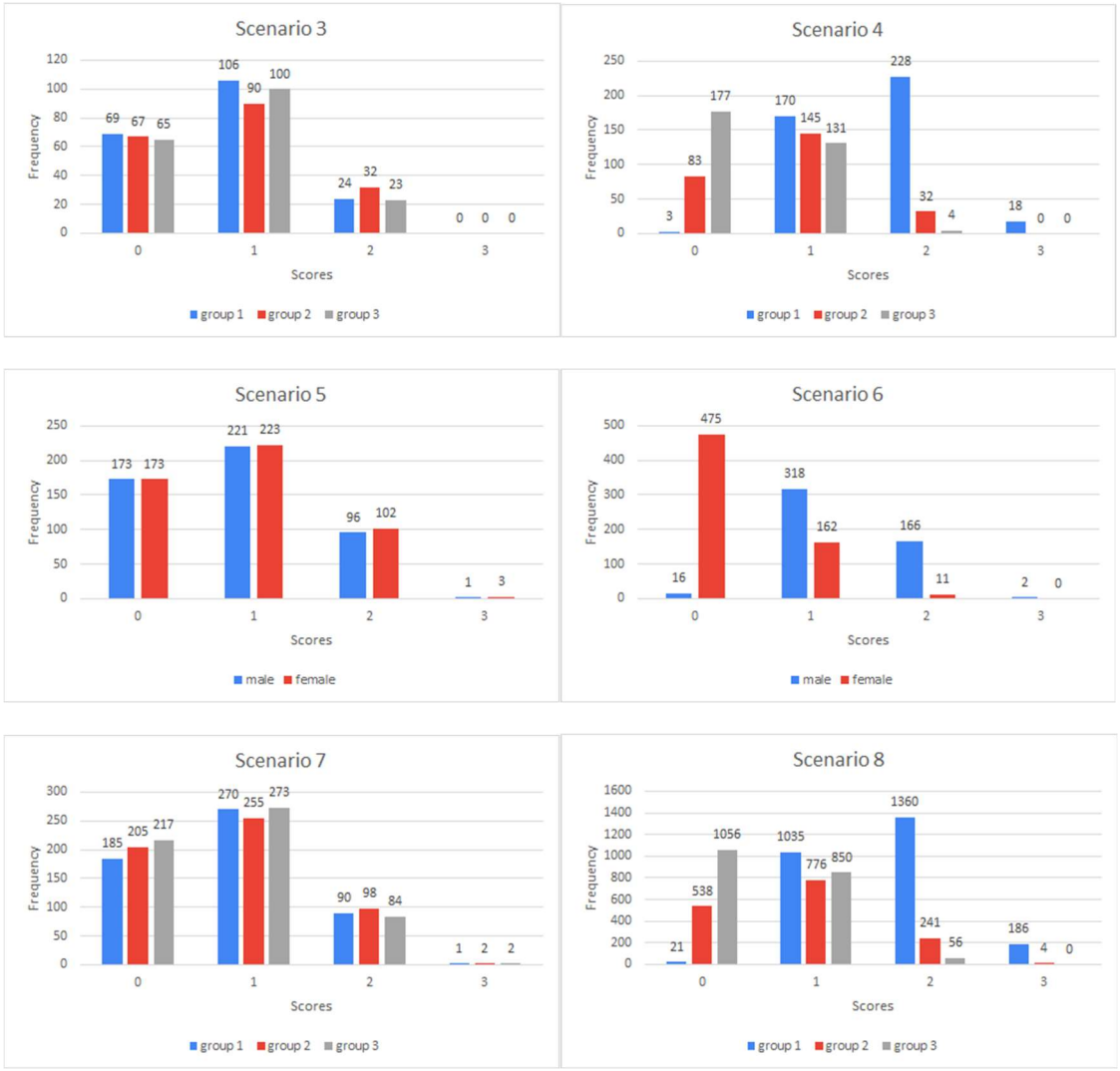


Fig. 2. The distribution of partial weight for experiments with 13 inputs.

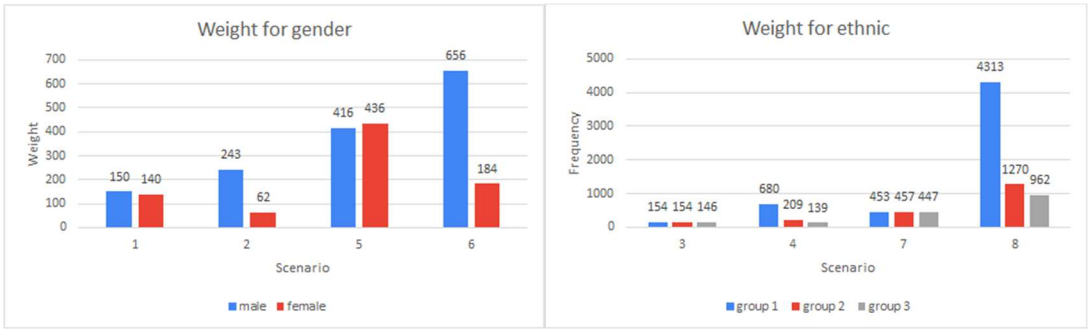


Fig. 3. The distribution of global weight for different gender (left) and ethnicity (right) for different scenarios.

Table 3. Distribution of the maximum difference for different scenarios.

Scenario	Bias	Gender		(%)	Scenario	Bias	Ethnicity			(%)
		Male (%)	Female (%)				group 1 (%)	group 2 (%)	group 3 (%)	
1	No	51.7	48.3	3.4	3	No	33.9	33.9	32.2	1.7
2	Yes	79.7	20.3	59.4	4	Yes	66.2	20.3	13.5	52.7
5	No	48.8	51.2	2.4	7	No	33.4	33.7	32.9	0.8
6	Yes	78.1	21.9	56.2	8	Yes	65.9	19.4	14.7	51.2

Partial Weight

The distribution of partial weight for each of the experiments is shown in Fig. 3.1.

As we can see, for unbiased experiments Scenario 1 and Scenario 5, the distribution of $gender(0)$and $gender(1)$is almost the same. By contrast, for Scenario 2 and Scenario 6, $gender(0)$appears in high scores with higher frequency while $gender(1)$appears mostly in low scores. As an example, $PW_{gender(0)}(scores(2))$ is more than 15 times more than $PW_{gender(1)}(scores(2))$ in both scenarios.

Moreover, gender information is not the only sensitive information that algorithms can extract from the data. By the comparisons of Scenario 3 (Scenario 7) with Scenario 4 (Scenario 8) in Fig. 3.1, we show that the decision made by the automatic recruitment system is also highly influenced by the ethnicity information.

Global Weight

We calculate the global weights for different values of gender/ethnicity and present the distribution of these global weights in Fig. 3.2.

As shown in left part of Fig. 3.2, in Scenario 1 and Scenario 5, the global weight of $gender(0)$(male) and $gender(1)$ (female) are almost the same, while for the Scenario 2 and Scenario 6, those gaps are huge. The phenomenon exists in the scenarios test for ethnicity.

To better understand the bias of the results, we induce the concept of the maximum difference (Δ). The results are presented in Table 3.3. In Scenarios 1 and 3, where the datasets are unbiased, we have almost no difference in the percentage of global weight. On

the other hand, in Scenarios 2 and 4, the impact of the bias is notorious, being larger with differences of 59.4% and 56.2%. The situation is held for ethnicity cases.

3.3.2.2. Analysis 2

For this analysis approach, we start from some statistical information of the logic programs learned by GULA-PRIDE. Then to answer the second problem, we show 1) the increment of the presence for each input after being normalized; 2) the percentage of the absolute increment for each input from biased experiments to its corresponding unbiased ones. The first one is shown by the comparison of the normalized percentage of each input, and the second one is shown by the distribution of a variable calculated from two corresponding programs.

For two corresponding programs P_1 and P_2, the frequency of attribute a in P_1 is defined as $freq_{P_1}(a)$. The normalized percentage for input i is:

$$NP_{p_1}(a) = freq_{P_1}(a) / \sum_{x \in input} freq_{P_1}(x),$$

and the percentage of the absolute increment for each input from unbiased experiments to its corresponding biased ones is defined as:

$$AIP_{P_1,P_2}(a) = (freq_{P_1}(x) - freq_{P_2}(x))/freq_{P_2}(x).$$

For the consideration of gender, we use all the experiments In Scenario 1, 2, 5, 6, totally 44 individual experiments. For our cases, P_1 is the individual experiment when we calculate NP_{p_1}, and P_1, P_2 are the biased experiment and its corresponding biased one when we calculate AIP_{P_1,P_2}.

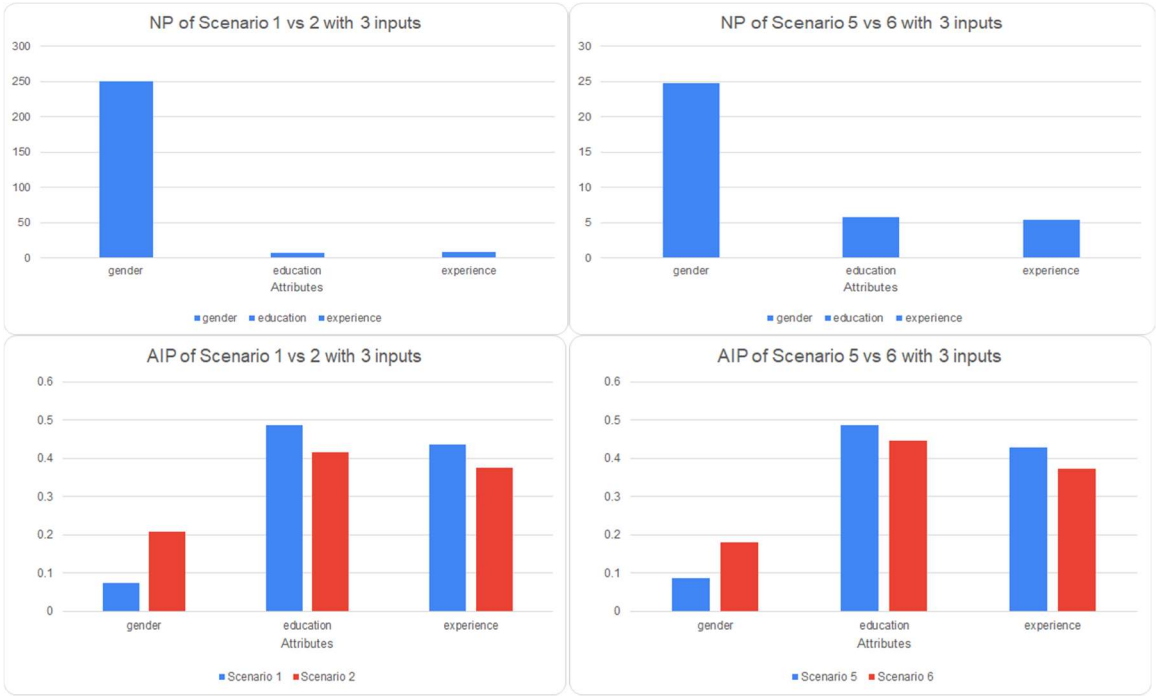


Fig. 4. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 3 inputs.

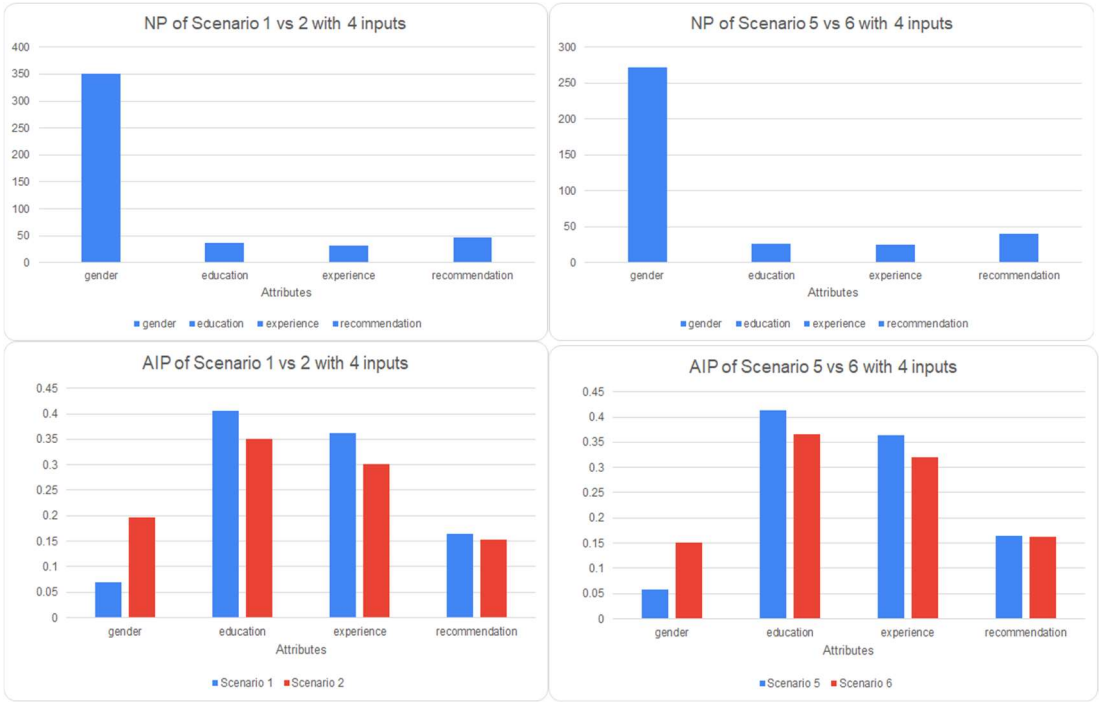


Fig. 5. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 4 inputs.

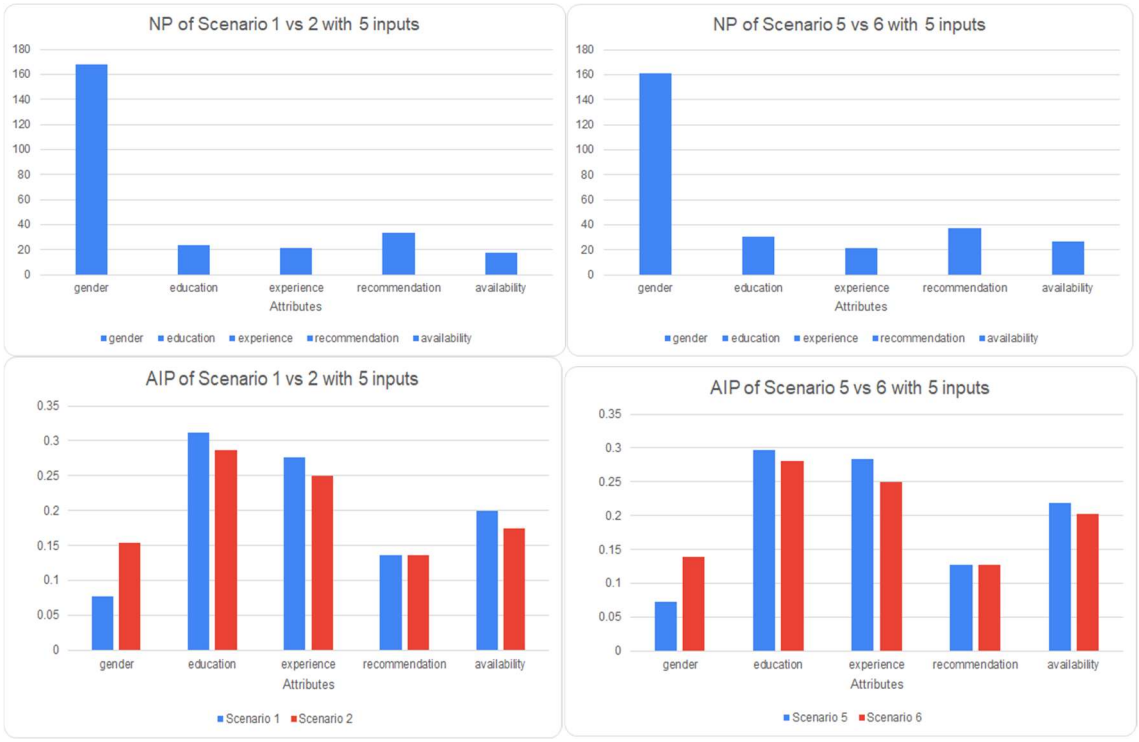


Fig. 6. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 5 inputs.

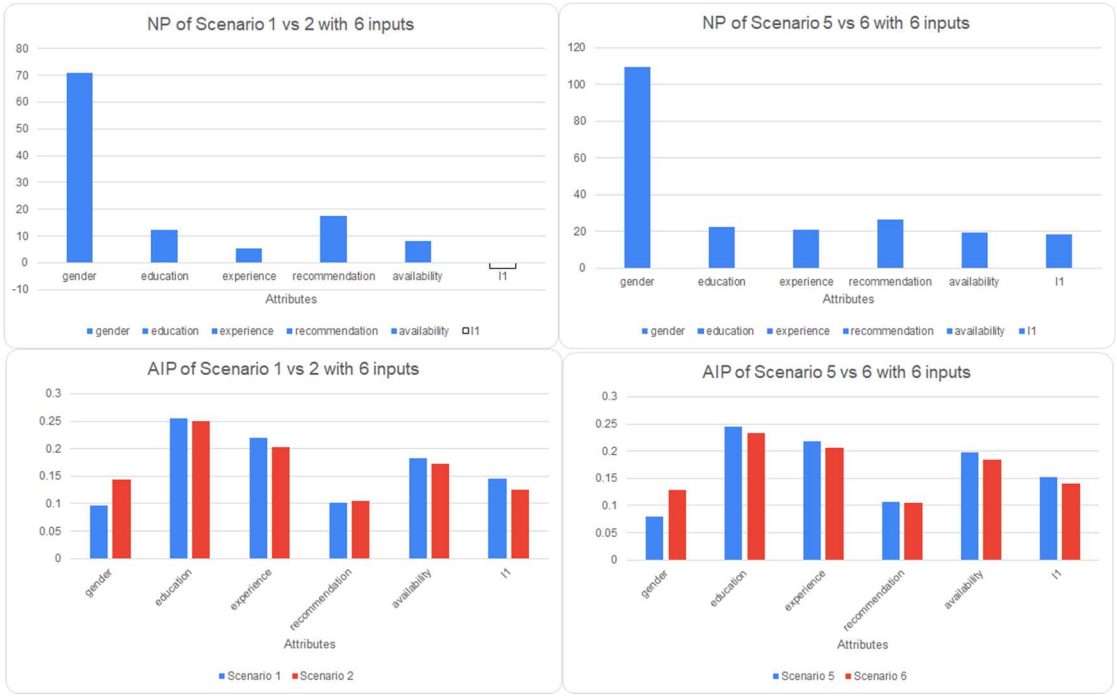


Fig. 7. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 6 inputs.

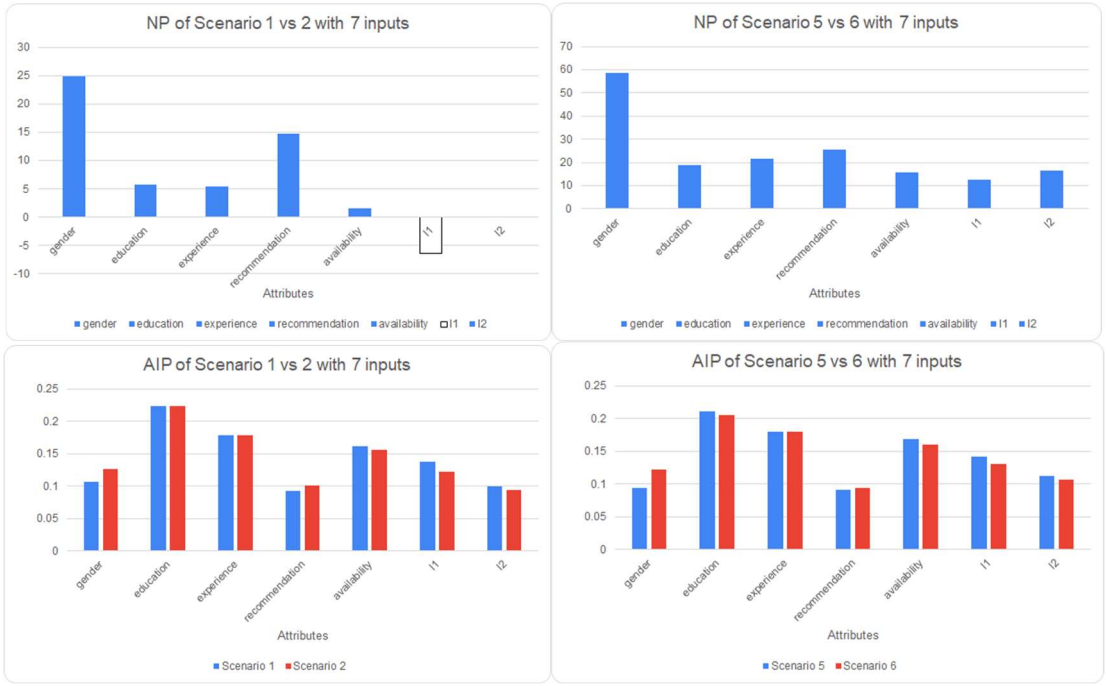


Fig. 8. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 7 inputs.

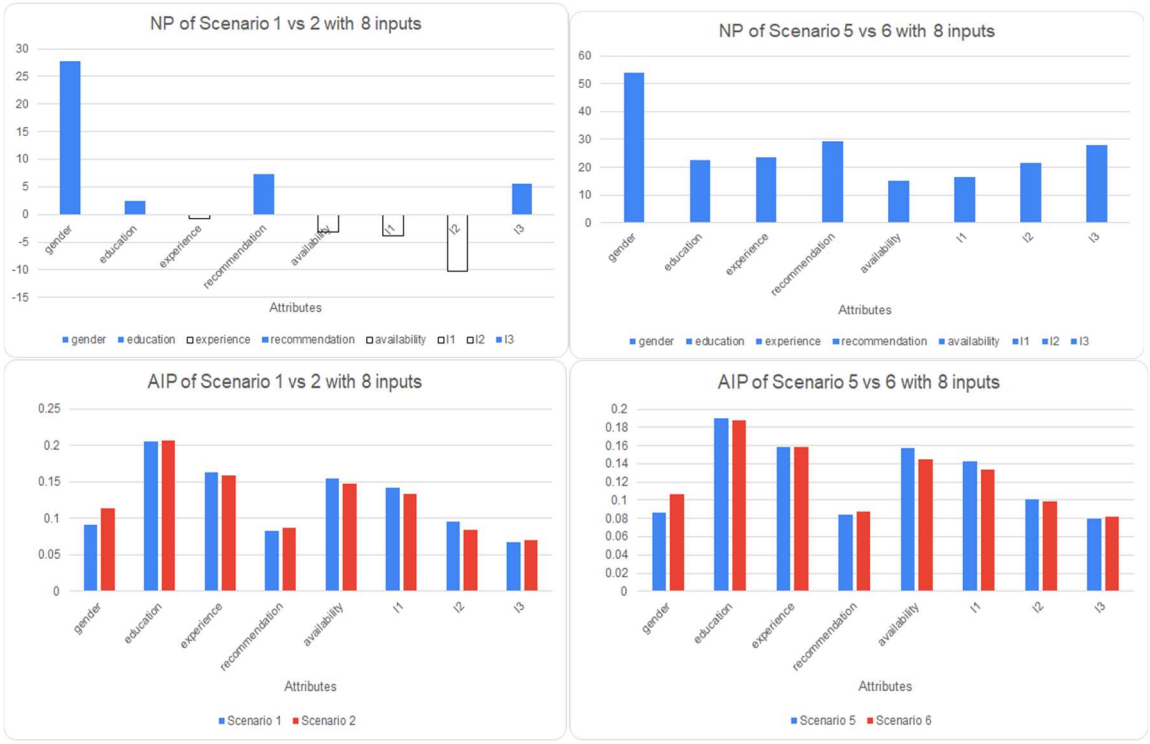


Fig. 9. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 8 inputs.

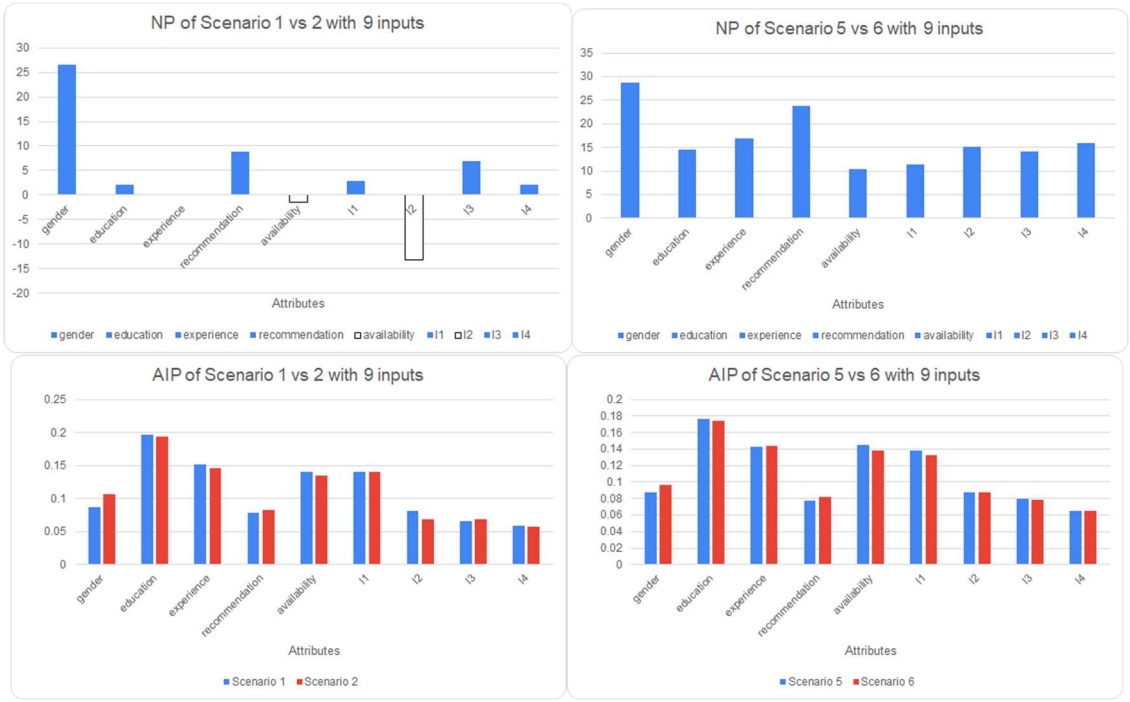


Fig. 10. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 9 inputs.

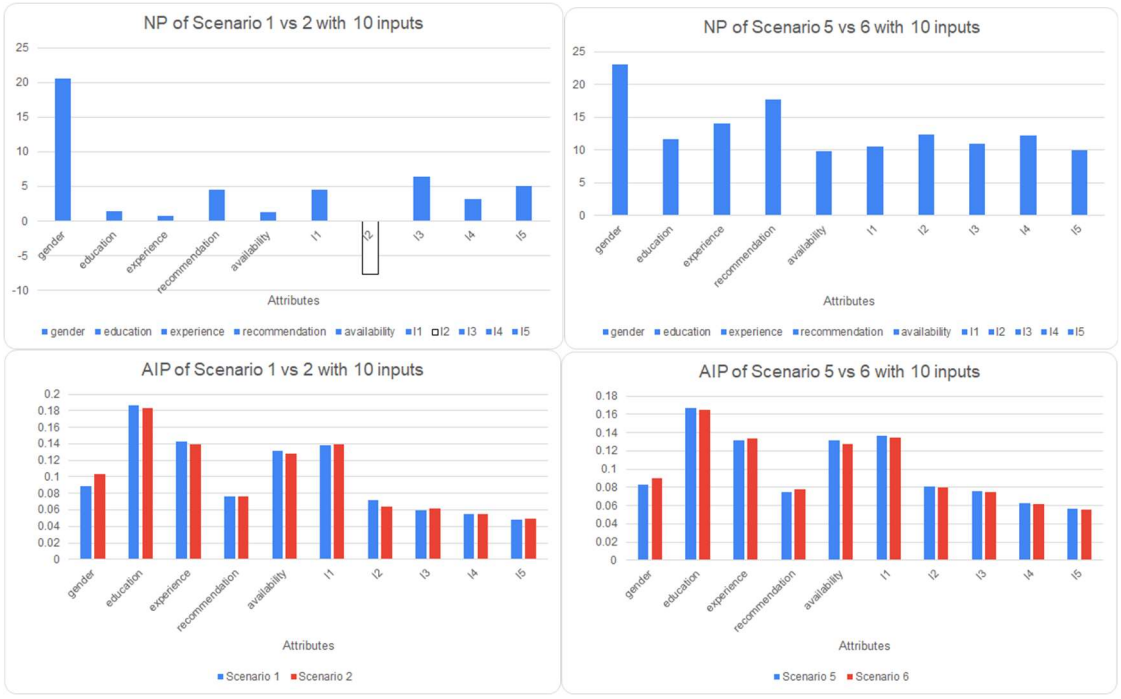


Fig. 11. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 10 inputs.

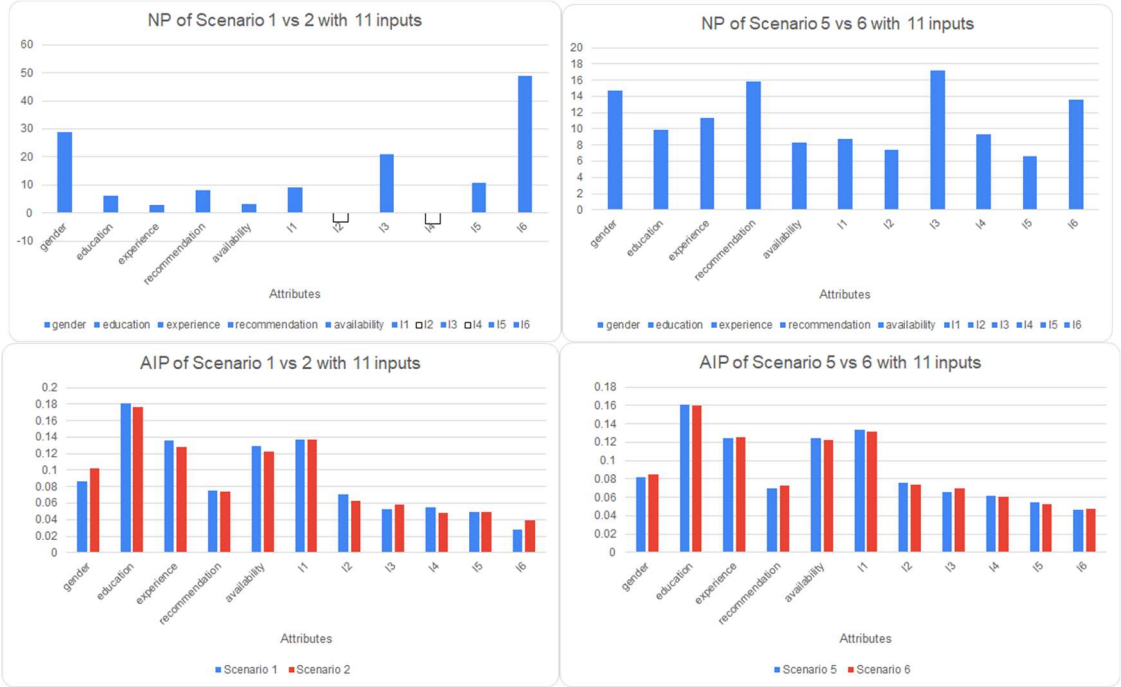


Fig. 12. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 11 inputs.

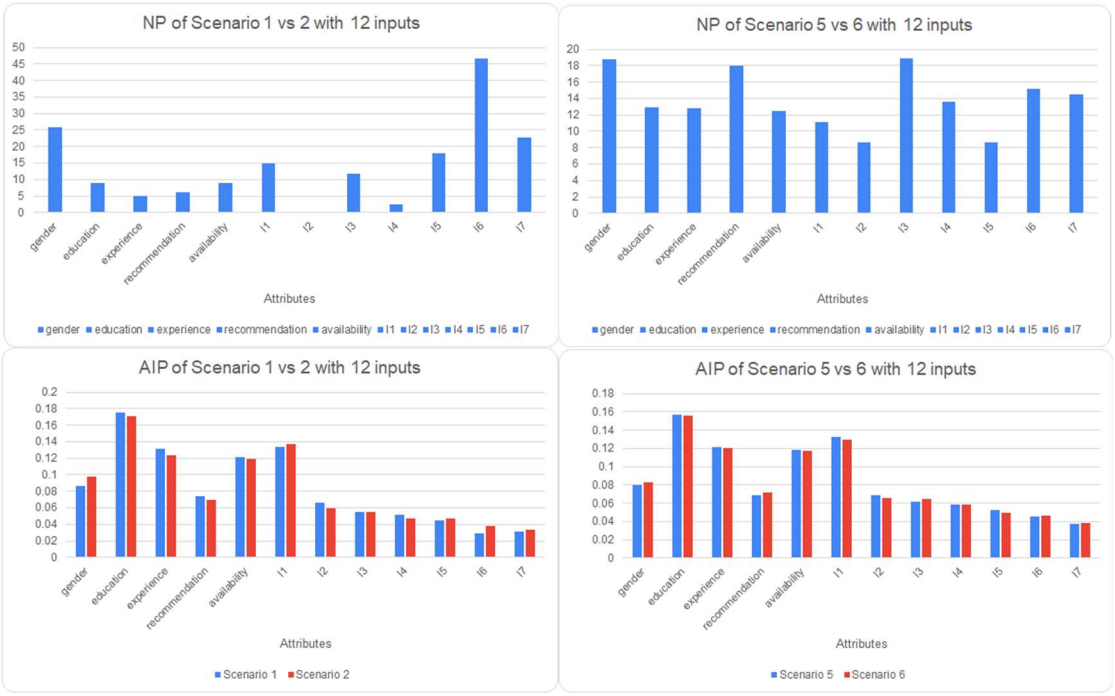


Fig. 13. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 12 inputs.

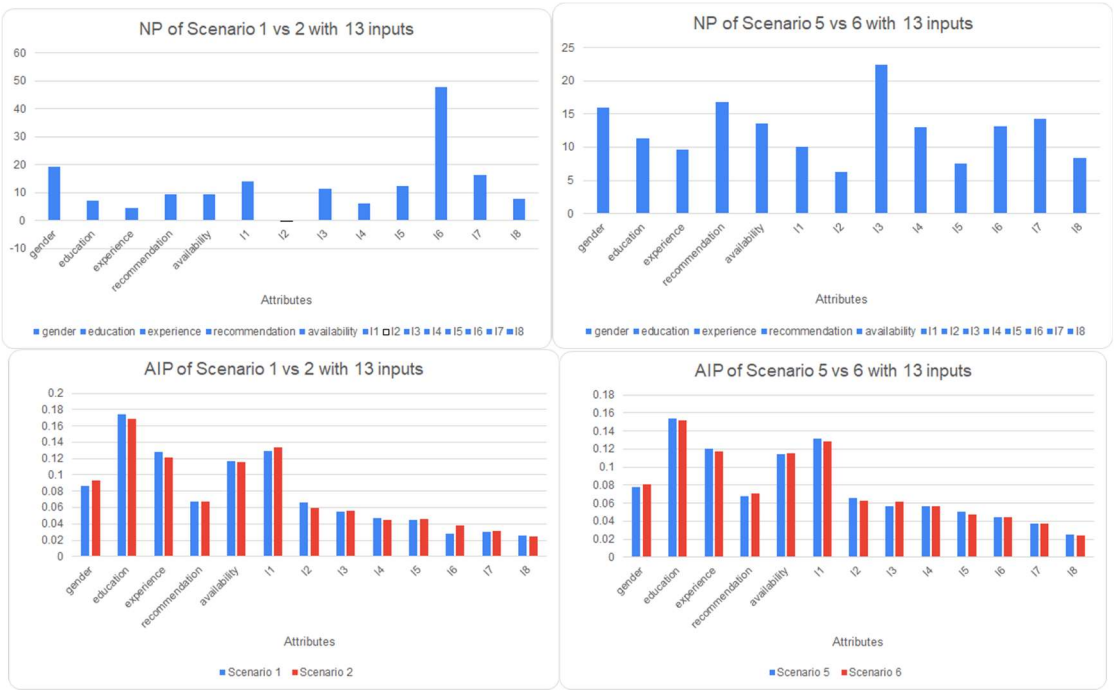


Fig. 14. Distributions of NP and AIP of Scenario 1 vs 2 (left) and Scenario 5 vs 6 (right) with 3 inputs.

As we explained before, two kinds of graphs are presented in Fig. 3.3 to Fig. 3.13.

Absolute Increment Percentage

The upper two graphs show the absolute increment of each input from unbiased logic programs to biased ones. These graphs allow us to compare the increment among different inputs.

From the vertical comparison of these graphs, we can find that, for all cases, the absolute increment percentage of attributes Gender grows. and this is more obvious when the number of inputs is less (more than 150% when only three inputs). This is because when there is less input, less bias is induced, the influence of Gender weakens. We can find that some of the attributes (language6, language3) tend to have a large increment percentage, we suppose that the reason is because these attributes are not individual and related to gender or ethnicity. This needs more experiments to verify. It can be seen also that the relative relevance of these attributes in the scores is quite small. Under these circumstances it is possible that small "real" changes are amplified. We think we should consider these variations more as noise than as data.

From the horizontal comparison of the two graphs in every figure, we find that the distribution of absolute increment percentage of different attributes becomes more evenly. Also, this shows again that these logic programs are just the logically equivalent versions of the data.

The distribution of absolute increment percentage for Ethnic is almost the same as Gender except that, in the horizontal comparison, the distributions are almost the same regardless of the number of inputs.

Normalized Percentage

The other two graphs put together the normalized presence of each input of unbiased with the biased ones to compare. With these graphs we can compare the relevance of each input's normalized percentage and see how each input changes between the biased and unbiased ones. From which we can conclude that Gender/Ethnic is not the most relevant attribute for the total score. It has been verified by all the cases that the rank of the attributes is: education, language1, availability, experience, gender, recommendation, language2, language3, language4, language5, language6, language7, language8. For ethnic, the relevance of ethnic is similar to language1. These ranks are reasonable to the experience of practice.

It is worthy to be mentioned that, for all the experiments with 3-12 inputs, we do not recalculate the scores. The main reason is that, for our case, the purpose is to try to catch the bias from gender/ethnic. Less inputs helps to amplify the bias of Gender/Ethnic. This point is also verified by our experiments.

3.3.2.3. Conclusions

From the two analysis approaches described in previous subsections, the answer for the second question is also positive.

From the statistical information of the logic programs learned by GULA-PRIDE, the preference of a high score is different for different values of Gender/Ethnic. To be specific, gender(0) > gender(1) and ethnic(0) > ethnic(1) > ethnic(2). What's more, the bias increases the relevance of Gender/Ethnic.

The graphical study of the presence of each attribute in the conditions of the rule exhibits these conclusions about the process (when comparing biased and unbiased scores)

- Attributes used to bias (gender and ethnicity) are not the most relevant for the scores. This fact has been confirmed with the author of the datasets.

- When biasing by gender (idem. ethnicity) gender (idem. ethnicity) is the attribute that always (in the complete sequence of experiments) increases its presence and most of the time it is the one that increases the most.
- There are a couple of attributes that increase its presence in the conditions of the rules when using all the input attributes but their relevance on the total score is small. This suggests that further experiments are needed to consider this increment as noise or to find a hidden relationship among them and the attributes used to bias the scores.

To remind also that the structure of the experiments performed is designed in this way because we decided to see if gender (idem. ethnicity) could wholly explain the scores. This is not the case. We decided to use a sequence of increasing number of attributes in the datasets without recomputing the scores. The accuracy of the program learnt by GULA-PRIDE started around 80% (with 3 attributes) and got the 100% with 12 attributes. This shows that almost all the attributes contribute to the final score.

4. CONCLUSIONS AND FURTHER RESEARCH LINES

This chapter reviews the contents of the previous chapters to briefly summarize the achievements.

This research is to explore the feasibility of providing an explanation for statistical machine learning approaches using GULA-PRIDE, and to what extent can GULA-PRIDE extract the bias of the unfair statistical machine learning approaches. Our conclusion can also be divided by two questions.

- Is it possible for GULA-PRIDE to explain what the deep learning process (automatic recruitment system) does?

The answer to this question is positive. With the help of GULA-PRIDE, we could finally get a logic program from the input and output of the process in a reasonable time for each scenario. The rules of these logic programs show the relations between inputs and output, which provide an explanation for the deep learning process. What's more, the high accuracy also shows that these logic programs are credible. And one of the most important characteristics: the theoretical basics of GULA-PRIDE ensure that the programs learnt are logically equivalent to the systems for the data used and that the conditions of each rule are minimal. These kinds of guarantees are crucial for explainable AI.

- Is GULA-PRIDE able to catch the bias of the deep learning process (automatic recruitment system)?

The answer to this question is also positive. We have two dimensions to support our point, which corresponds to two different approaches.

1. From the statistical information (PW, GW and Δ) of the logic programs learned by GULA-PRIDE, the preference of a high score is different for different values of Gender/Ethnic. To be specific, gender(0) > gender(1) and ethnic(0) > ethnic(1) > ethnic(2), and the upper (lower) bound for biased (unbiased) scenarios could be as small (large) as 3.4% (51.2%).

2. We define NP and AIP of a logic program, from which we conclude that: 1) Although Gender/Ethnic are not the most relevant attributes to scores, the bias increases the relevance of Gender/Ethnic compared to the unbiased scenarios; 2) It has been verified that the rank of the attributes is: education, language1, availability, experience, gender, recommendation, language2, language3, language4, language5, language6, language7, language8. For ethnic, the relevance of ethnic is similar to language1. 3) when comparing programs learnt from biased and unbiased datasets it is obvious the increment of occurrences of the attributes used to bias (gender/ethnicity) in all the scenarios.

5. FURTHER RESEARCH LINES

In this chapter, we provide a global view of completed work and some instructions for future works.

- Improve the GULA-PRIDE to make it be empowered with prediction capabilities.

  As we explained in CHAPTER 3, the logic programs learned by GULA-PRIDE are just logically equivalent versions of the data rather than prediction models. So, the behavior of the logic program is similar to an over-fitted statistical machine learning model. GULA-PRIDE is aware of this circumstance and it is known that for adding prediction capabilities to the programs learnt these have to be post processed using domain dependent information. For the future work, we would like to propose a systematic way to modify the rules in this scenario.

- Apply GULA-PRIDE to other practical systems to reveal the inner relationship of these systems.

  GULA-PRIDE is rather a general algorithm which could be used to analyze any system to reveal the inner relation of the system. For example, we could apply GULA-PRIDE to a corpus to reveal some relations to verify the theories from generative linguistics.

- Increase the interpretability of the system

  We have observed that, although declarative, the theoretical guarantees of GULA-PRIDE tend to produce rules in some way complex.
  We would like to consider mechanisms to clarify them.
  One of the approaches that could be interesting are ILP systems based on first order logic. Their ability to invent recursive predicates could improve this aspect.
  The use of these systems could be also advisable to get predictive explanations because these systems are machine learning focused not only focused on explanations. The goal of GULA-PRIDE of generating a digital twin of a black box (for the inputs provided) suggests to use them to get direct explanations.

REFERENCES

[Acien et al., 2018] Acien, A., Morales, A., Vera-Rodriguez, R., Bartolome, I., & Fierrez, J. (2018, November). Measuring the gender and ethnicity bias in deep models for face recognition. In Iberoamerican Congress on Pattern Recognition (pp. 584-593). Springer, Cham.

[Ali et al., 2019] Ali, M., Sapiezynski, P., Bogen, M., Korolova, A., Mislove, A., & Rieke, A. (2019). Discrimination through optimization: How Facebook's ad delivery can lead to skewed outcomes. arXiv preprint arXiv:1904.02095.

[Alireza, 2013] Logic-based machine learning using a bounded hypothesis space: the lattice structure, refinement operators and a genetic algorithm approach. Alireza Tamaddoni Nezhad Imperial College London, August 2013.

[Cropper, 2017] Efficiently learning efficient programs A. Cropper PhD thesis, Imperial College London, 2017.

[Cropper et al., 2019] Cropper, A., & Muggleton, S. H. (2019). Learning efficient logic programs. Machine Learning, 108(7), 1063-1083.

[Cropper et al., 2020] Cropper, Andrew and Dumančić, Sebastijan and Muggleton, Stephen H. Turning 30: New Ideas in Inductive Logic Programming Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. International Joint Conferences on Artificial Intelligence Organization. Christian Bessiere. 4833--4839. 2020 https://doi.org/10.24963/ijcai.2020/673 .

[DeRaedt et al 2018] DeepProbLog: Neural Probabilistic Logic Programming, Robin Manhaeve and Sebastijan Dumančić and Angelika Kimmig and Thomas Demeester and Luc De Raedt, 2018, arXiv1805.10872.

[Dai et al., 2015] Dai, W. Z., Muggleton, S. H., & Zhou, Z. H. (2015). Logical Vision: Meta-Interpretive Learning for Simple Geometrical Concepts. In ILP (Late Breaking Papers) (pp. 1-16).

[Drozdowski et al., 2020] Drozdowski, P., Rathgeb, C., Dantcheva, A., Damer, N., & Busch, C. (2020). Demographic bias in biometrics: A survey on an emerging challenge. IEEE Transactions on Technology and Society.

[Evans 2017] Learning Explanatory Rules from Noisy Data, Richard Evans and Edward Grefenstette, 2017, arXiv1711.04574.

[Fiérrez et al., 2017a] Multiple classifiers in biometrics. part 1: Fundamentals and review Fiérrez, Julián; Morales, Aythami; Vera-Rodriguez, Rubén; Camacho, David 2017-12-22.

[Fiérrez et al., 2017b] Multiple classifiers in biometrics. Part 2: Trends and challenges Fiérrez, Julián; Morales, Aythami; Vera-Rodriguez, Rubén; Camacho, David 2017-11-22.

[Gebser et al., 2012] Answer Set Solving in Practice. Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub Synthesis Lectures on Artificial Intelligence

and Machine Learning, Morgan and Claypool December 2012, 238 pages, doi:10.2200/S00457ED1V01Y201211AIM019.

[Goodman et al., 2016] Goodman, B., & Flaxman, S. (2016, June). EU regulations on algorithmic decision-making and a "right to explanation". In ICML workshop on human interpretability in machine learning (WHI 2016), New York, NY. http://arxiv. org/abs/1606.08813 v1.

[He et al., 2016] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[Herrera et al., 2020] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, Francisco Herrera, Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion, Volume 58, 2020, Pages 82-115.

[Inoue et al., 2014] Inoue, K., Ribeiro, T., & Sakama, C. (2014). Learning from interpretation transition. Machine Learning, 94(1), 51-79.

[Kanehira et al., 2019] Kanehira, A., & Harada, T. (2019). Learning to explain with complemental examples. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 8603-8611).

[Katayama, 2005] S. Katayama. Systematic search for lambda expressions. In Sixth Symposium on Trends in Functional Programming, pages 195–205, 2005.

[Kodiyan, 2019] Kodiyan, A. A. (2019). An overview of ethical issues in using AI systems in hiring with a case study of Amazon's AI based hiring tool.

[Koza, 92] Koza J.R., (1992) Genetic programming. MIT Press. Cambridge, MA.

[Lloyd, 87] Lloyd, J. W. (1987). Foundations of Logic Programming. (2nd edition). Springer-Verlag.

[Martínez et al., 2015] Martínez, D., Ribeiro, T., Inoue, K., Alenyà Ribas, G., & Torras, C. (2015). Learning probabilistic action models from interpretation transitions. In Proceedings of the Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015) (pp. 1-14).

[Martínez et al., 2016] Martínez, D., Alenyà Ribas, G., Torras, C., Ribeiro, T., & Inoue, K. (2016). Learning relational dynamics of stochastic domains for planning. In Proceedings of the 26th International Conference on Automated Planning and Scheduling, ICAPS 2016 (pp. 235-243).

[Michie, 1988] Michie, D. (1988). Machine learning in the next five years. In Proceedings of the third European working session on learning (pp. 107–122). Pitman.

[Molnar, 2020] Christoph Molnar. Interpretable Machine Learning. A Guide for Making Black Box Models Explainable 2020-02-28 Edited by learnpub.

[Molnar et al., 2020] Molnar, Christoph & König, Gunnar & Herbinger, Julia & Freiesleben, Timo & Dandl, Susanne & Scholbeck, Christian & Casalicchio, Giuseppe & Grosse-Wentrup, Moritz & Bischl, Bernd. (2020). Pitfalls to Avoid when Interpreting Machine Learning Models.

[Morales et al., 2019] Morales, A., Fierrez, J., & Vera-Rodriguez, R. (2019). SensitiveNets: Learning agnostic representations with application to face recognition. arXiv preprint arXiv:1902.00334.

[Muggleton, 1991] Muggleton, S. (1991). Inductive logic programming. New generation computing, 8(4), 295-318.

[Muggleton et al., 2014] Muggleton, S. H., Lin, D., Pahlavi, N., & Tamaddoni-Nezhad, A. (2014). Meta-interpretive learning: application to grammatical inference. Machine learning, 94(1), 25-49.

[Muggleton et al.,, 2018a] Muggleton, S. H., Schmid, U., Zeller, C., Tamaddoni-Nezhad, A., & Besold, T. (2018). Ultra-Strong Machine Learning: comprehensibility of programs learned with ILP. Machine Learning, 107(7), 1119-1140.

[Muggleton et al., 2018b] Muggleton, S., Dai, W. Z., Sammut, C., Tamaddoni-Nezhad, A., Wen, J., & Zhou, Z. H. (2018). Meta-interpretive learning from noisy images. Machine Learning, 107(7), 1097-1118.

[Mukund et al., 2019a] Si, Xujie & Raghothaman, Mukund & Heo, Kihong & Naik, Mayur. (2019). Synthesizing Datalog Programs using Numerical Relaxation. 6117-6124. 10.24963/ijcai.2019/847.

[Mukund et al., 1019b] Raghothaman, Mukund & Mendelson, Jonathan & Zhao, David & Naik, Mayur & Scholz, Bernhard. (2019). Provenance-guided synthesis of Datalog programs. Proceedings of the ACM on Programming Languages. 4. 1-27. 10.1145/3371130.

[Ortega et al., 2007] Alfonso Ortega, Marina de la Cruz, Manuel Alfonseca: Christiansen Grammar Evolution: Grammatical Evolution With Semantics. IEEE Trans. Evol. Comput. 11(1): 77-90 (2007).

[Peña et al., 2020] Peña, A., Serna, I., Morales, A., & Fierrez, J. (2020). Bias in Multimodal AI: Testbed for Fair Automatic Recruitment. arXiv preprint arXiv:2004.07173.

[Ribeiro, 2015] Ribeiro, T. (2015). Studies on Learning Dynamics of Systems from State Transitions.

[Ribeiro et al., 2018] Ribeiro, T., Folschette, M., Magnin, M., Roux, O., & Inoue, K. (2018, September). Learning dynamics with synchronous, asynchronous and general semantics. In International Conference on Inductive Logic Programming (pp. 118-140). Springer, Cham.

[Ryan et al., 2003] Michael O'Neill and Conor Ryan. 2003. Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, USA.

[Schmid et al., 2016] Schmid, U., Zeller, C., Besold, T., Tamaddoni-Nezhad, A., & Muggleton, S. H. (2017). How does predicate invention affect human comprehensibility? In A. Russo & J. Cussens (Eds.) Proceedings of the 26th international conference on inductive logic programming (ILP 2016, September 4th–6th, London). Springer.

[Senior, 2012] Senior, A., Vanhoucke, V., Nguyen, P., & Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition. IEEE Signal processing magazine.

[Tome et al., 2013] Tome, P., Fierrez, J., Vera-Rodriguez, R., & Ramos, D. (2013). Identification using face regions: Application and assessment in forensic scenarios. Forensic science international, 233(1-3), 75-83.

[Tome et al., 2014] Tome, P., Fierrez, J., Vera-Rodriguez, R., & Nixon, M. S. (2014). Soft biometrics and their application in person recognition at a distance. IEEE Transactions on information forensics and security, 9(3), 464-475.

[Tome et al., 2015] Tome, P., Vera-Rodriguez, R., Fierrez, J., & Ortega-Garcia, J. (2015). Facial soft biometric features for forensic face recognition. Forensic science international, 257, 271-284.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

APPENDIX A

In Appendix A, we show the output of the GULA-PRIDE for Scenario 1 with 3 inputs.

PRIDE output:
FEATURE gender 0 1
FEATURE education 0 1 2 3 4 5
FEATURE experience 0 1 2 3 4
TARGET scores 0 1 2 3

scores(0) :- education(1), experience(1).
scores(0) :- education(1), experience(0).
scores(0) :- education(0), experience(1).
scores(0) :- education(0), experience(2).
scores(0) :- education(0), experience(0).
scores(0) :- education(2), experience(0).
scores(0) :- education(2), experience(1).
scores(0) :- education(1), experience(2).
scores(1) :- gender(0), education(0).
scores(1) :- experience(1).
scores(1) :- education(1), experience(2).
scores(1) :- education(0), experience(2).
scores(1) :- education(3), experience(2).
scores(1) :- education(2).
scores(1) :- gender(1), education(1).
scores(1) :- education(3), experience(0).
scores(1) :- education(1), experience(3).
scores(1) :- education(4), experience(0).
scores(1) :- education(0), experience(3).
scores(1) :- gender(1), education(4).
scores(1) :- education(4), experience(2).
scores(1) :- gender(1), education(5), experience(0).
scores(2) :- education(3), experience(2).
scores(2) :- gender(0), education(5).
scores(2) :- education(2), experience(2).
scores(2) :- education(4), experience(1).
scores(2) :- education(5), experience(2).
scores(2) :- education(2), experience(1).
scores(2) :- education(4), experience(2).
scores(2) :- education(3), experience(1).
scores(2) :- education(3), experience(3).
scores(2) :- education(1), experience(2).
scores(2) :- education(5), experience(1).
scores(2) :- education(1), experience(3).
scores(2) :- education(2), experience(3).

```
scores(2) :- education(4), experience(3).
scores(2) :- education(5), experience(3).
scores(3) :- education(5), experience(2).
scores(3) :- gender(1), education(5), experience(3).
scores(3) :- education(4), experience(3).

Timing:  0:00:00.104093

    DEPRECATED
    Model accuracy: 80.21527777777777%
```